DOCUMENTS

Client SDK und User-Exits

Manuel Tromm

DOPaK 2018

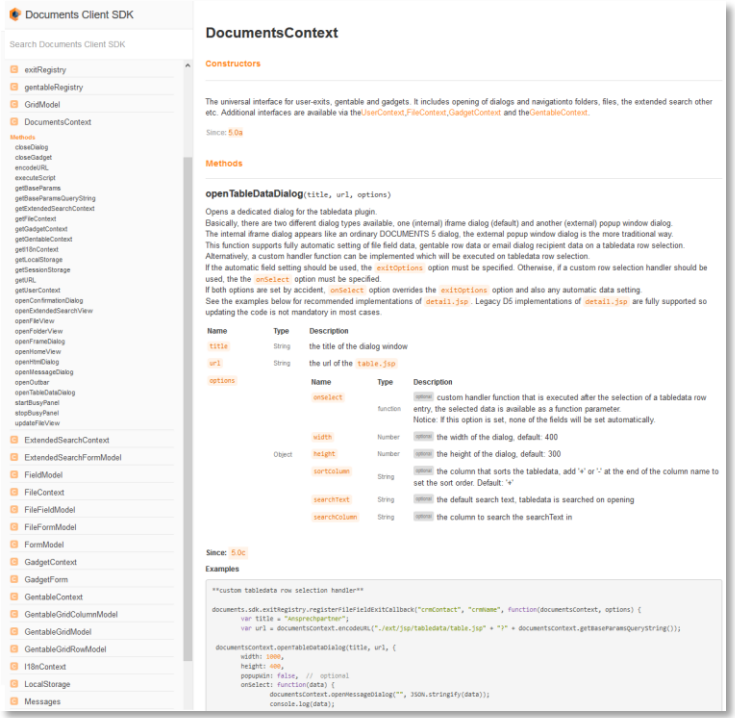
Client SDK – Einleitung

Worum geht es beim Client SDK?

- Zugriff & Manipulation der D5 WebClient GUI
  - Berechnungen & Formularlogik
  - Constraints & Plausibilität
  - Layout
- nicht direkt auf Server realisierbare Funktionalität
- User-Exits, TableData, Gentable und Gadgets

Hilfe

- Client SDK API Dokumentation
- HowTos und programmierte Beispiele



Client SDK – Rückblick

Callbacks

- implementieren eigene Funktionalität (JavaScript)
- bewirken Änderung des Standardverhaltens je Situation
- haben vollen Zugriff auf Client SDK API
- werden von D5 WebClient bei bestimmen Ereignissen ausgeführt

Registrierung

- verknüpft Callback mit Ereignis
- erfolgt mit Funktionen der `exitRegistry` oder `gentableRegistry`

Client SDK – Rückblick

DocumentsContext

- repräsentiert universale Schnittstelle zu Client SDK API
- Zugriff auf div. Sub-Contexts (`UserContext`, `FileContext`, `GentableContext` etc.)
- Callbacks werden mit Funktionen aus Contexts implementiert
- verfügbar in User-Exits, Gentable, Gadgets, …

Externe Ressourcen

- externe Verzeichnisstruktur, separiert von D5 Installation
- Lagerort für eigenen Custom-Code, AddOns, Bibliotheken, Konfigurationen
- Konfiguration (Mapping) in Tomcat erfolgt mit `documents.xml`
- `script.jsp` importiert Skript-Ressourcen in D5 WebClient

Neues von TableData

Neue Funktion `openTableDataDialog()`

- elementare Vereinfachung zur bisherigen Variante
- Anzeige im aktuellen D5 Dialog-Layout
- automatisches Setzen von Feldern
- Suchspalte, Suchbegriff und Sortierung
- individuelle Parameter konfigurierbar
- automatische Codierung aller Parameter
- in Mappe, Gentable, Erweiterte Suche, Script-Parameter-Dialog, …
- einfache Migration (`table.jsp` kompatibel, ggf. minimale Anpassung in `detail.jsp`)

Neues von TableData

Beispiel: `openTableDataDialog()`

Aufruf mit Titel, URL und Optionen

Dialog intern/extern

Suchspalte, -begriff und Sortierung

Custom-Parameter

empfangene Daten aus `detail.jsp` verarbeiten

```
1  var title = "Artikel",
2      url = documentsContext.encodeURL("./ext/jsp/product/table.jsp" + "...");
3
4  documentsContext.openTableDataDialog(title, url, {
5      width: 800,
6      height: 400,
7      popupWin: false,  //  default: false
8
9      searchColumn: "Artikelbezeichnung",
10     searchText: "Coffee Arabica",
11
12     sortColumn: "Artikel-Nr",
13
14     params : {
15         supplierId : "86305"
16     },
17
18     onSelect: function(data) {
19         //  set selected table row data
20     }
21 });
```

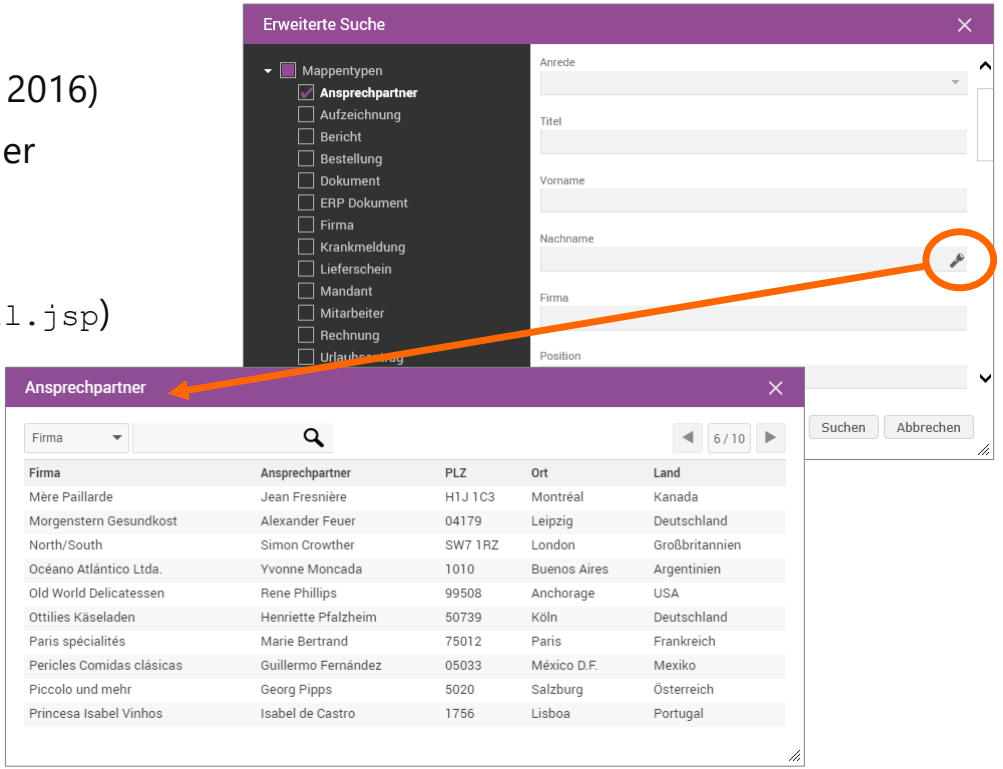
Exits in erweiterter Suche

Beispiel: TableData

- analog zum Beispiel in Mappe (DOPaK 2016)
- Import von externen Daten in Suchfelder

Umsetzung

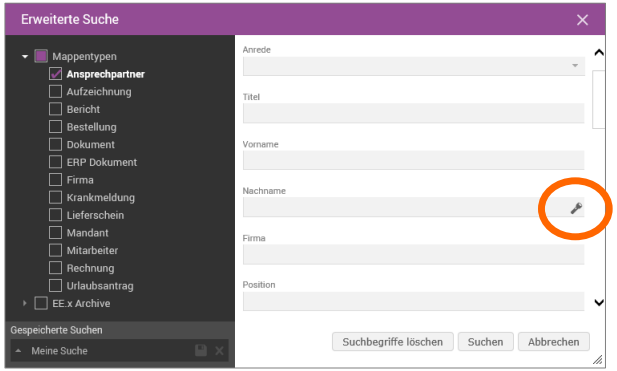
- nutzt bekannte JSPs (`table.jsp`, `detail.jsp`)
- Callback 1 bereitet Exit auf Feld vor
- Callback 2 auf Feld öffnet Tabelle



Exits in erweiterter Suche

Beispiel: TableData (Callback 1)

- Ausführung vor Anzeige der Suchfelder
- untersucht ausgewählte Suchquellen des Baums
- untersucht verfügbare Felder des Formulars
- Feld „Nachname“ erhält ggf. Exit mit Button



Registrierung

Suchquellen / Felder

Mappentyp suchen

Feld suchen

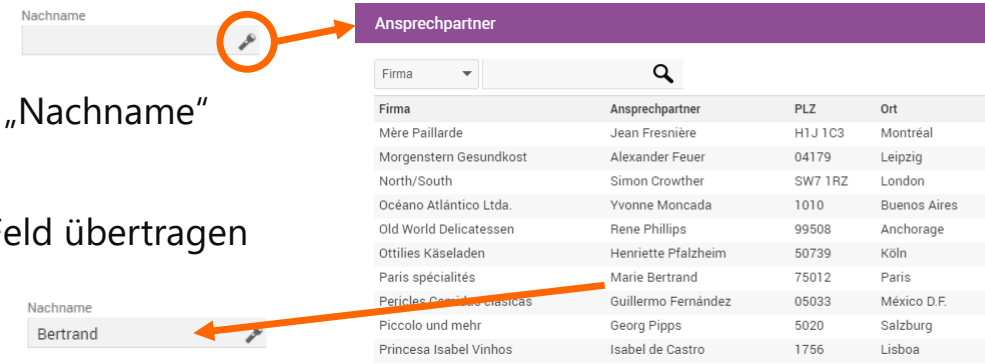
Exit auf Feld setzen

```
1  documents.sdk.exitRegistry.registerSearchExitCallback(
2      "ExtendedSearch.afterSetModelData", function(documentsContext, options) {
3
4      var extSearchContext = documentsContext.getExtendedSearchContext(),
5          searchSources = extSearchContext.getSelectedSearchSourceNames(),
6          fields = extSearchContext.getSearchFormModel().getSearchFields(), field;
7
8      if(searchSources.indexOf("crmContact") >= 0) {
9
10         if(field = fields.get("crmName")) {
11
12             field.setExit({ type: "button", image: "entypo:flashlight", tooltip: "..." });
13         }
14     }
15 });
```

Exits in erweiterter Suche

Beispiel: TableData (Callback 2)

- Ausführung bei Klick auf Exit-Button bei „Nachname“
- öffnet Dialog mit `openTableData()`
- gewählter Eintrag (`detail.jsp`) wird in Feld übertragen



Registrierung

Feld „Nachname“

TableData-Dialog

Feldwert setzen

```
20 documents.sdk.exitRegistry.registerSearchFieldExitCallback(
21     "crmName", function(documentsContext, options) {
22
23     var title = "Ansprechpartner",
24         url = documentsContext.encodeURL("./ext/jsp/tabledata/table.jsp" + "..."),
25         field = options.searchField;
26
27     documentsContext.openTableDataDialog(title, url, {
28         width: 800,
29         height: 400,
30         onSelect: function(data) {
31             field.setValue(data.crmName);
32         }
33     });
34 });
```

detail.jsp

```
{
    crmAccountNumber: "PARIS",
    crmName: "Bertrand",
    crmFirstName: "Marie",
    ...
}
```

Exit-Callbacks in erweiterter Suche

- Callbacks bei Ereignissen in Feldern (z.B. `focusout`, `change`, `button`, …)
  - `exitRegistry.registerSearchFieldExitCallback(fieldName, fn)`
- Callbacks bei Ereignissen im Dialog
  - `exitRegistry.registerSearchExitCallback(event, fn)`

Beispiele für Callback-Ereignisse im Dialog	**Beschreibung**	**Abbruch bei return false;**
`ExtendedSearch.afterSetModelData`	nach Empfang der Daten	-
`ExtendedSearch.afterRenderSearchForm`	nach Rendern der Suchfelder	-
`ExtendedSearch.beforeExecuteSearch`	vor Start der Suche	ja
`ExtendedSearch.afterExecuteSearch`	nach Ende der Suche	-
`ExtendedSearch.afterToggleSearchSource`	nach Wechsel der Suchquelle	-
`ExtendedSearch.afterToggleSearchMask`	nach Wechsel der Suchmaske	-
`ExtendedSearch.afterToggleHitListMask`	nach Wechsel der Trefferliste	-

Exits in Skript-Parameter-Dialogen

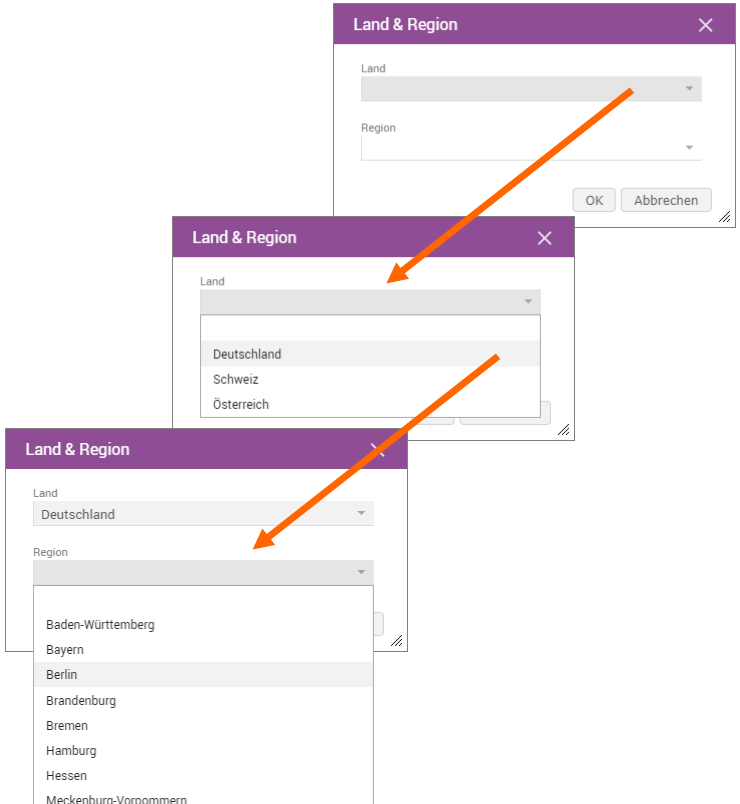
Beispiel: Dynamische DropDown-Liste

Anforderungen

- Feld „Land“ hält Länder
- Feld „Region“ hält Regionen je Land
- Änderung von Land passt Regionen automatisch an
- Feld „Region“ ist gesperrt, falls kein Land gewählt

Umsetzung

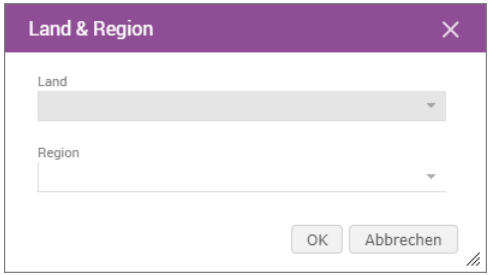
- Callback 1 bei Start des Skripts bereitet Exit vor
- Callback 2 auf Feld „Land“ ruft Skript auf
- Skript auf Server liefert Regionen aus



Exits in Skript-Parameter-Dialogen

Beispiel: Dynamische DropDown-Liste (Callback 1)

- Ausführung vor Anzeige der Felder
- Feld „Land“ erhält Exit bei Wertänderung
- Feld „Region“ ist initial schreibgeschützt



Registrierung

alle Felder

Exit auf Feld „Land“

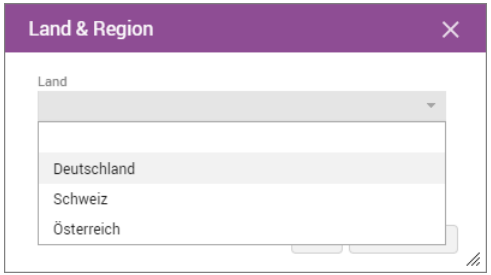
Feld „Region“ sperren

```
1  documents.sdk.exitRegistry.registerScriptParameterExitCallback(
2      "country-region", "ScriptParameter.afterSetModelData", function(documentsContext, options) {
3
4      var fields = options.scriptForm.getFields();
5
6      var field1 = fields.get("crmCountry");
7      field1.setExit({ type: "change" });
8
9      var field2 = fields.get("crmRegion");
10     field2.setGuiReadonly(true);
11 });
```

Exits in Skript-Parameter-Dialogen

Beispiel: Dynamische DropDown-Liste (Callback 2)

- Ausführung bei Änderung von Land
- Skript-Aufruf (synchron) mit Feldwert aus „Land“
- setzt aus Skript empfangene Werte in Feld „Region“



Registrierung

Feld „Land“ auslesen

Script mit Parameter `country` ausführen

Feld „Region“ setzen

```
15 documents.sdk.exitRegistry.registerScriptParameterFieldExitCallback(
16     "country-region", "crmCountry", function(documentsContext, options) {
17
18     var fields = options.scriptForm.getFields(),
19         country = options.scriptFieldValue;
20
21     var regions = documentsContext.executeScript("countryScript", { "country" : country });
22
23     var field = fields.get("crmRegion");
24     field.setEnumValues(regions);
25     field.setGuiReadonly(regions === "");
26 });
```

Exits in Skript-Parameter-Dialogen

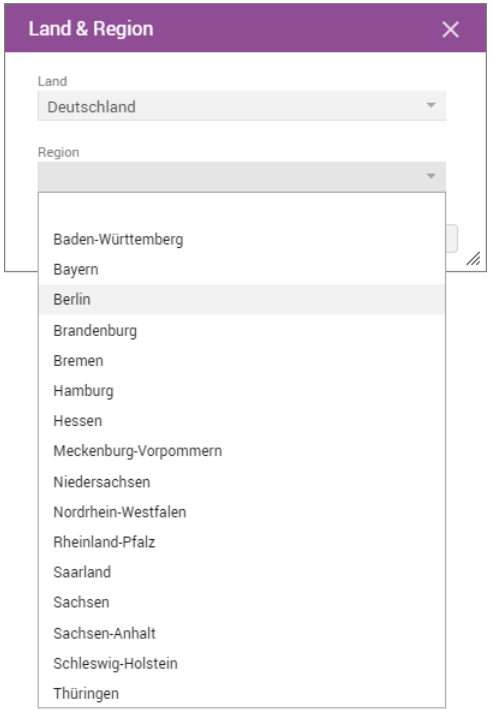
Beispiel: Dynamische DropDown-Liste (Skript)

- `countryScript` gibt Regionen je Land zurück
- liefert keine Regionen, falls kein Land gewählt ist
- Skript-Parameter `country` wird von Skript-Aufruf übermittelt

```
var regions = {
    "Germany": ["Baden-Württemberg", "Bayern", "Berlin", "Brandenburg", "Bremen", "..."],
    "Austria": ["Burgenland", "Kärnten", "Niederösterreich", "Oberösterreich", "..."],
    "Switzerland": ["Zürich", "Bern", "Luzern", "Uri", "Schwyz", "Obwalden", "..."]
}

if(!country) {
    return "";
}

return regions[country];
```



Exit-Callbacks in Skript-Parameter-Dialogen

- Callbacks bei Ereignissen in Feld (z.B. `focusout`, `change`, `button`, …)
  - `exitRegistry.registerScriptParameterFieldExitCallback(scriptName, fieldName, fn)`
- Callbacks bei Ereignissen in Dialog
  - `exitRegistry.registerScriptParameterExitCallback(scriptName, event, fn)`

Beispiele für Callback-Ereignisse in Dialog	**Beschreibung**	**Abbruch bei return false;**
`ScriptParameter.afterSetModelData`	nach Empfang der Daten	-
`ScriptParameter.beforeRender`	vor Rendern des Dialogs	-
`ScriptParameter.afterRender`	nach Rendern des Dialogs	-
`ScriptParameter.beforeExecuteScript`	vor Start des Skripts	ja

Asynchrone Skripte mit JSON

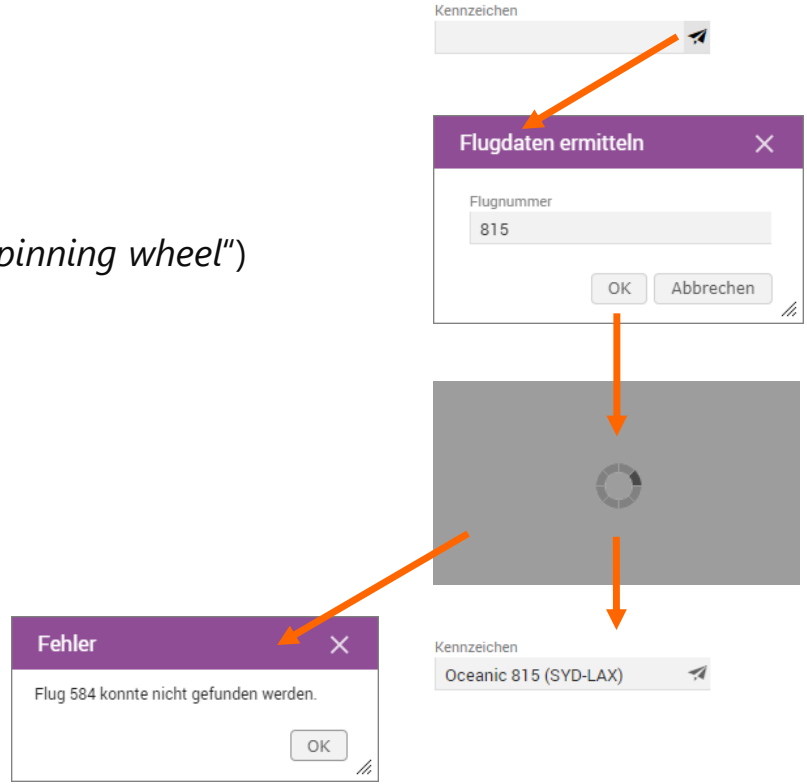
Beispiel: Flugdaten ermitteln

Anforderungen

- Flugdaten mit Flugnummer suchen
- Mappe sperren während laufender Suchanfrage („*spinning wheel*“)
- Flug-Kennzeichen in Feld übertragen, falls OK
- Fehlermeldung, falls nicht gefunden

Lösung

- Callback ruft Skript mit Parameterdialog
- Skript sucht und gibt Flugdatensatz zurück
- Datenformat: JSON



Asynchrone Skripte mit JSON

Beispiel: Flugdaten ermitteln (Skript)

Fall: Fehler

Flugdatensatz „Oceanic 815“

Fall: OK

Ausgabe (JSON)

```
1  if(flightNumber !== "815") {
2      context.errorMessage = "Flug " + flightNumber + " konnte nicht gefunden werden.";
3      return -1;
4  }
5
6  var data = {
7      airline: "Oceanic",
8      number: 815,
9      departure: {
10         IATA: "SYD",
11         time: "2004-09-22 14:55",
12         city: "Sydney"
13     },
14     arrival: {
15         IATA: "LAX",
16         time: "2004-09-23 10:42",
17         city: "Los Angeles"
18     },
19     numbers: [4, 8, 15, 16, 23, 42]
20 }
21
22 return JSON.stringify(data);
```

Skript-Parameter

Name	Bezeichnung	Typ
flightNumber	Flugnummer	String

```
{"airline":"Oceanic","number":815,"departure":{"IATA":"SYD","time":"2004-09-22 14:55","city":"Sydney"},"ar
```

Asynchrone Skripte mit JSON

Beispiel: Flugdaten ermitteln (Callback)

Registrierung

spinning wheel ein

Skript-Aufruf (async)

Fall: OK

JSON Daten

spinning wheel aus

Fall: Fehler

```
1  documents.sdk.exitRegistry.registerFileFieldExitCallback(
2      "flightRecord", "flightLabel", function(documentsContext, options) {
3
4      var scriptOptions = {
5          async: true,
6          useScriptParameterDialog: true
7      };
8
9      documentsContext.startBusyPanel("MainFile");
10
11     var promise = documentsContext.executeScript("flight-data", { }, scriptOptions);
12
13     promise.then(function(jsonData) {
14
15         var data = JSON.parse(jsonData),
16             field = options.fileForm.getFieldByName("flightLabel"),
17             value = data.airline + " " + data.number + "...";
18
19         field.setValue(value);
20         documentsContext.stopBusyPanel("MainFile");
21     })
22     .catch(function(error) {
23         documentsContext.stopBusyPanel("MainFile");
24     });
25 });
```

Exit-Callbacks in Gentable

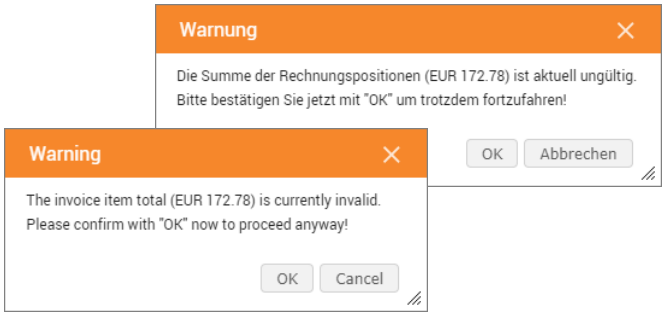
- Callbacks bei Ereignissen in Feld oder Button (z.B. in `<event>` oder `<function>`)
  - `gentableRegistry.registerCallback(gentableDefinitionName, event, fn)`
- Callbacks bei allg. Ereignissen in Gentable
  - `gentableRegistry.registerCallback(gentableDefinitionName, event, fn)`

Beispiele für Callback-Ereignisse in Gentable	**Beschreibung**	**Abbruch bei return false;**
`Gentable.beforeRender`	vor Rendern der Tabelle	-
`Gentable.afterRender`	nach Rendern der Tabelle	-
`Gentable.beforeStore`	vor Speichern der Tabelle	ja
`Gentable.afterStore`	nach Speichern der Tabelle	-
`Gentable.afterRowSelect`	nach Selektion der Zeile	-

Internationalisierung (I18n)

I18nContext

- einfacher Zugriff auf eigene Nachrichten in aktueller Sprache des Benutzers
- lokalisierte Bezeichner in eigenen Properties-Dateien je Sprache (z.B. `custom_de.properties`, `custom_en.properties`, ...)
- Unterstützung für Wildcards
- Lagerort: Externe Ressource (z.B. `/ext/resource/` oder `/ext/i18n/`)




```
1 var messages = documentsContext.getI18nContext().getMessages("custom"),
2     title = messages.get("title"),
3     description = messages.get("description", null, ["172.78"]);
4
5 documentsContext.openConfirmationDialog(title, description, onOK, onCancel);
```

```
25 title = Warnung
26 description = Die Summe der Rechnungspositionen (EUR {1}) ist aktuell ungültig...
```

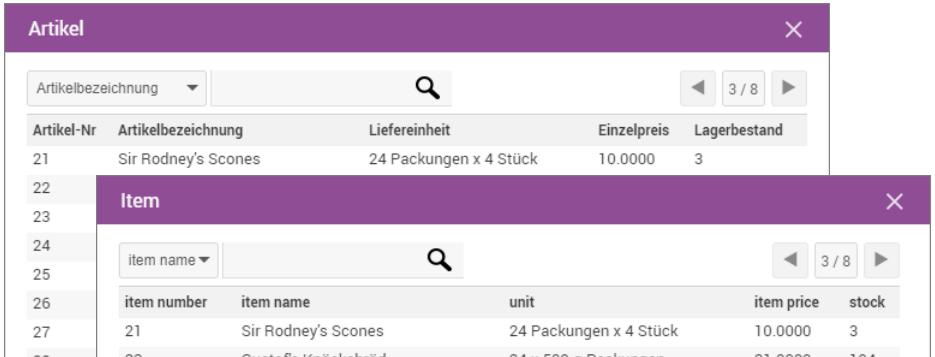
```
25 title = Warning
26 description = The invoice item total (EUR {1}) is currently invalid...
```

Internationalisierung (I18n)

TableData

- lokalisierte Bezeichner in Properties-Dateien je Sprache (`TableData_de.properties`, `TableData_en.properties`, ...)
- Lagerort: Externe Ressource (z.B. `/ext/resource/` oder `/ext/i18n/`)

```
52 <tab:tableconfig id="item" />
53     ...
54     <tab:headline>itemno,itemname,unit,itemprice,stock</tab:headline>
55     ...
56 </tab:tableconfig>
```




```
16 itemno = Artikel-Nr
17 itemname = Artikelbezeichnung
18 unit = Liefereinheit
19 itemprice = Einzelpreis
20 stock = Lagerbestand
```

```
16 itemno = item number
17 itemname = item name
18 unit = unit
19 itemprice = item price
20 stock = stock
```

Internationalisierung (I18n)

Gentable Feldtyp NUMBER

- Daten sind unabhängig von der aktuellen Sprache
- sprachabhängige Anzeige von Dezimal- und Tausendertrennzeichen
- Formatierung der Nachkommastellen für die Anzeige
- Dezimaltrennzeichen und Präzision der Nachkommastellen für die Daten
- Berechnungen in Exits immer in JavaScript-Datentyp `Number`

Nr	Artikelnr.	Artikeltext	Menge	Stückpreis	Betrag	KST	Konto
1	83212408	BMW Engine Oil LL04 5W30	12	166,60	1.999,20		

no	item number	description	amount	unit price	total	cost unit	account
1	83212408	BMW Engine Oil LL04 5W30	12	166.60	1,999.20		

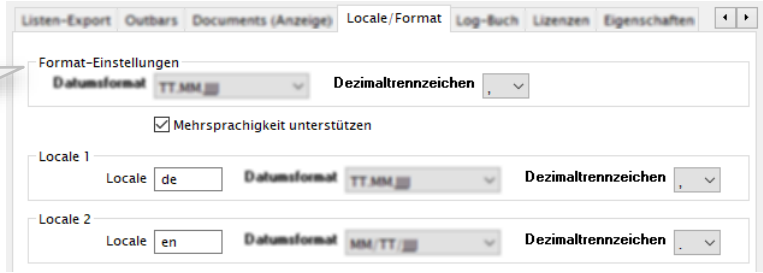
Internationalisierung (I18n)

Gentable Feldtyp NUMBER

```
<field number="5">
    <label>de:Betrag;en:total</label>
    <title>total</title>
    <type>NUMBER</type>
    <width>80</width>
    <numberFormat>n.2</numberFormat>
    <editable>true</editable>
</field>
```

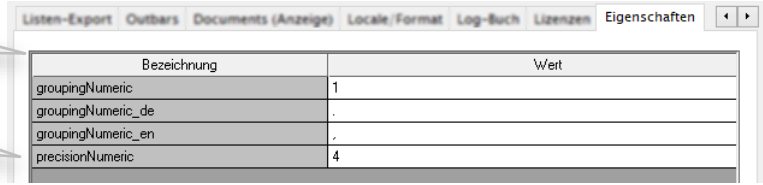
Dezimaltrenner (Daten & Anzeige)

Nachkommastellen

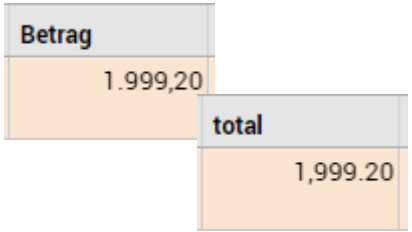


Tausendertrenner

Präzision (Daten)



Daten-XML




```
<table><tr>...<td title="total">1999,2000</td>...</tr></table>
```

otris software
vereinfacht Verantwortung.

Vielen Dank!

Manuel Tromm

www.otris.de

otris software AG
Königswall 21
44137 Dortmund